TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Vladislav Ogorodnik 184913IAIB

VEEBILIIDES RELATSIOONILISTE ANDMETE TEISENDAMISEKS RDF KUJULE

Bakalaureusetöö

Juhendaja: Priit Järv PhD
Kaasjuhendaja: Marko Vendelin PhD

Tallinn 2021

Autodeklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Vladislav Ogorodnik
14.06.2021

Annotatsioon

Enamus teadusandmeid hoiustatakse relatsioonilistes andmebaasides. Nende teadusandmete publitseerimine kooskõlas FAIR printsiipidega on keerukas ja aeganõudev protsess. See keerukus toob endaga kaasa probleemi, et teadlased ei ole alati valmis kulutama palju aega oma andmete publitseerimiseks formaadis, mis tagab nende parima leitavuse, kättesaadavuse, koostalitusvõime ja korduvkasutatavuse. Parandades teadusandmete publitseerimise ökosüsteemi suudame parandada avastuste ja innovatsioonide levikut, mis viib kogukonna arengut edasi.

Lõputöö eesmärgiks on luua tarkvara, mis võimaldab annoteerida relatsioonilises andmebaasis asuvad teadusandmed ja transformeerida need RDF kujule. Tarkvara peab automatiseerima aeganõudvat käsitsi protsessi ning olema kasutatav domeeni eksperdi poolt, kellel puuduvad sügavad teadmised SQL-st ja semantilisest veebist.

Oma panusena kirjeldasime FAIR printsiipidega kooskõlas teadusandmete publitseerimise käsitsi protsessi ja kaardistasime käsitsi protsessi lihtsustava vabatarkvara. Samuti arendasime RDB-RDF transformatsiooni tegeva prototüübi potentsiaalse edasise töö aluseks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 39 leheküljel, 4 peatükki, 11 joonist, 3 tabelit.

Abstract

Web Application for Converting Relational Data to RDF

Most scientific data is stored in relational databases. Scientific data publishing in accordance with FAIR principles is a complex and time-consuming process. This complexity poses the problem that researchers are not always willing to spend a lot of time publishing their data in a format that ensures their best findability, accessibility, interoperability and reusability. By improving the ecosystem for publishing scientific data, we can improve knowledge discovery that drive community development forward.

The aim of the dissertation is to create software that allows annotating scientific data, stored in a relational database, and mapping it to RDF. The software must automate a time-consuming manual process and be usable by a domain expert with no deep knowledge of SQL and the Semantic Web.

As a contribution, we described the manual process of publishing scientific data in accordance with the FAIR principles and mapped free software that manually simplifies the process. We also developed a prototype for the RDB to RDF mapping as a basis for potential further work.

The thesis is in Estonian and contains 39 pages of text, 4 chapters, 11 figures, 3 tables.

Lühendite ja mõistete sõnastik

API	*Application programming interface* rakendusliides
CellML	XML-põhine märgistuskeel matemaatiliste mudelite kirjeldamiseks
CURIE	*Compact URI* kompaktne ühtne ressursiidentifikaator
D2RQ	Virtuaalselt relatsioonandmebaasidele juurdepääsu süsteem
FAIR principles	*Findable, Accessible, Interoperable, Reusable principles* leitavuse, kättesaadavuse, koostalitusvõime ja korduvkasutatavuse põhimõtted
IRI	*Internationalized Resource Identifier* spetsiaalne ressursiidentifikaator
JDBC	*Java Database Connectivity* Java andmebaaside ühenduvus
JSON	*JavaScript Object Notation* JavaScript objektide tähistus
MIREOT	*The Minimum Information to Reference an External Ontology Term* minimaalne teave välise ontoloogilise termini viitamiseks
MVP	*Minimum viable product* minimaalselt töötav toode
OBO	*Open Biomedical Ontologies* avatud biomeditsiinilised ontoloogiad
OWL	*Web Ontology Language* veebiontoloogia keel
REST	*Representational state transfer*
RDB	*Relational Database* relatsiooniline andmebaas
RDBMS	*Relational Database Management Systems* relatsioonandmebaaside haldussüsteemid
RDF	*Resource Description Framework* ressursikirjelduse raamistik
R2RML	Relatsioonilise andmebaasi ja RDF kaardistukeel
SBML	*Systems Biology Markup Language* süsteemibioloogia märgistuskeel
SPARQL	*Protocol and RDF Query Language* RDF päringu keel
SQL	*Structured Query Language* Struktureeritud päringu keel
TalTech	*Tallinn University of Technology* Tallinna Tehnikaülikool
URI	*Uniform Resource Identifier* ühtne ressursiidentifikaator

Sisukord

Joonised

Tabelid

1 Sissejuhatus

Bakalaureusetöö eesmärk on arendada TalTech süsteemibioloogia labori rakenduse *Kinetics* kõrvale tarkvara, mis parandab rakenduse tehtud katsete andmete FAIR printsiipide: leitavuse, kättesaadavuse, koostalitusvõime ja korduvkasutatavuse jälgimist [1]. FAIR on kõrgtaseme põhimõtete kogum, mis kirjeldab andmetehalduse head tavad andmete säilitamise parandamiseks. Avatud teadusandmete levitamise eesmärk ei ole Süsteembioloogia labori keskne eesmärk, vaid see kattub kogu TalTechi üldise suunaga [2]. Praegune teadusandmete avaldamise digitaalne ökosüsteem ei võimalda saadud tulemustest maksimaalse kasu saamist, kuna enamasti on sobivate andmete leidmine, nende formaadi ja kasutusloa määramine ning automaatse importimise integreerimine raske. FAIR printsiibi abil saab parandada teadusandmete automaatset kättesaadavust masinate ja inimeste jaoks nende lihtsamaks taaskasutamiseks [4].

1.1 Probleemi püstitus

Bioloogilised mõõtmised hõlmavad sageli parameetrite mõõtmist aja, ruumi või sageduse funktsioonina. Hiljem, uuringu analüüsietapis, jagab teadlane salvestatud andmete jälje väiksemateks osadeks, analüüsib iga jaotist eraldi, leides keskmise või sobitades määratud funktsiooni ja kasutab analüüsi tulemusi uuringus. Selle uurimisvoo lihtsustamiseks on TalTech Süsteemibioloogia labor arendanud rakenduse *Kinetics*, mis võimaldab analüüsida mõõdetud parameetri sõltuvust ühest muust parameetrist, näiteks ajast või ruumist. Lisaks uurimisvoole võimaldab tarkvara andmete jälgi analüüsida viisil, mis tagab analüüsi korratavuse ja lihtsustab FAIR (leitavuse, kättesaadavuse, koostalitlusvõime ja korduvkasutatavuse) põhimõtete rakendamist sellistes uuringutes. Samal ajal lihtsustab see tavapärast andmeanalüüsi ja annab juurdepääsu analüüsitulemuste kiirele ülevaatele. Selleks toetab tarkvara algandmete lugemist, protokollis täpsustatud andmete töötlemist ja kõigi vahetulemuste salvestamist labori andmebaasi [1]. *Kinetics* rakendusel puudub sisseehitatud andmete annoteerimise ja eksportimise võimalus, mis ei võimalda kasutada tarkvara täielikult FAIR printsiibiga kooskõlas. Lisaks ei võimalda see andmete annoteerimist [3].

Bakalaureusetöö eesmärk on luua tarkvara, mis parandaks *Kinetics*-i vastavust FAIR printsiipidele. Loodav rakendus peab võimaldama kasutajal lisada andmebaasi andmetele annotatsioone ja eksportida andmeid standardiseeritud kujul, selleks et neid saaks importida teadusandme-

baasidesse. Andmed plaanitakse publitseerida FAIR printsiipe jälgivatesse teadusandmebaasidesse nagu *Seek4science* [5]. Kuna *Kinetics* on vabavara, siis on see potentsiaalselt kasutusele võetav ka teiste teaduslaborite poolt ning arendatavast rakendusest oleks kasu ka neile [3].

1.2 Töö panus ja lähtetingimused

Süsteembioloogia laboril on soov luua *Kinetics* rakenduse kõrvale tarkvara, mis võimaldaks *Kinetics*-i mugavamalt kasutada kooskõlas FAIR printsiipidega. Praegu nõuab uuringutulemuste publitseerimine kooskõlas FAIR printsiipidega palju käsitööd ning erinevate vahendite kasutusoskust. Arvestatavaks puuduseks on Süsteembioloogia labori kompetentsi ja teadmiste puudumine, kuidas ja mis vahenditega võiks jõuda püstitatud eesmärgini. Samuti ei ole laboril selget arusaamist, kuidas teha sama protsessi käsitsi. Enne soovitud tarkvara arendust on vaja teha korralik eelanalüüs, mille käigus täpsustada tarkvara nõuded. Analüüsi käigus selgus, et soovitud rakendust ei ole bakalaureusetöö skoobis võimalik arendada ning püstitatud eesmärk sai korrigeeritud. Uueks eesmärgiks sai arendada prototüüp analüüsi väljundi testimiseks.

Bakalaureusetöö on valminud koostöös Süsteembioloogia laboriga. Lõputöö kirjutamise vältel on toimunud iganädalsed koosolekud Marko Vendeliniga, süsteemibioloogia laboratooriumi vanemteaduri ja juhatajaga. Analüüsi faasis toimus laboratooriumi töö protsessidega tutvumine, jooksev tarkvara nõuete täpsustus ning analüüsi tulemuste arutelu. Analüüsi tulemuste arutlus aitas korrigeerida lõputöö nõudeid ning otsustada, mis suunas liikuda edasi. Arenduse faasis sai otsustatud kasutatavate tehnoloogiate üle, kokkulepitud graafilise liidese vaade ning arendatud rakenduse prototüüp. Arenduse faasi vältel toimusid iganädalased koosolekud uute arenduste demonstreerimiseks neile tagasiside saamiseks.

2 Analüüsi faas

2.1 Teadusandmete avaldamise digitaalse ökosüsteemi kitsaskohad

Mugav teadusandmete publitseerimine ja avaldatud andmete haldus on peamine vahend, mis viib teadmiste avastamiseni ja innovatsioonini. Andmete haldus on enamat kui eesmärk ise. See on peamine vahend, mis peab olema tagatud, et avastused saaksid levida ja olla taaskasutatava ning viia kogukonna arengut edasi. Kahjuks takistab olemasolev teadusandmete publitseerimise digitaalne ökosüsteem maksimaalse kasu saamist publitseeritud andmetest [4].

Praegune andmete publitseerimise ökosüsteem on liikumas eemale tsentraliseerimisest. See tähendab, et tekkinud on arvukaid üldotstarbelisi teadusandmebaase, mis aktsepteerivad laia valikut andmetüüpe, väga erinevates formaatides ning tavaliselt ei sea nad andmete publitseerijatele suuri piiranguid ehk puudub kindel standard. See viib selleni, et publitseerija ise vastutab andmeformaadi valiku eest. Samuti teadusandmebaaside rohkuse tõttu muutub andmeökosüsteem vähem tsentraliseerituks ning andmete vähese integreerituse ja mitmekesisuse tõttu süveneb publitseeritud andmete leitavuse ja taaskasutatavuse probleem [4].

Toome välja süsteembioloogia näitel võimaliku takistuse. Oletame, et teadlane on teostanud loomkatseid ning soovib võrrelda oma tulemusi teiste sarnaste katsetulemustega. Teadlane peab vastama järgnevatele küsimustele. Kui eksisteerib soovitud andmekogum, siis kus see võib olla publitseeritud ja mis otsingutööriistu kasutada. Soovitud otsing peaks filtreerima konkreetse loomaliigi järgi ja teiste teadlast huvitavate parameetrite järgi. Kui selline otsing on võimalik, siis tekivad täiendavad küsimused. Kas metaandmed on talletatud teadusandmebaasi? Mis formaadis on andmed? Kas antud formaat on hõlpsasti taaskasutatav koos teadlase enda andmetega? Millistel litsentsitingimustel ja kellele tuleks viidata, kui andmeid taaskasutatakse [4]?

Eelnevad küsimused illustreerivad takistusi, millega puutub kokku teadlane oma igapäevatöös. Kirjeldatud teadusandmete avaldamise probleemid ei ole tingitud sellest, et puuduksid vajalikud tehnoloogiad vaid selle tõttu, et publitseerijad ei pööra vajaliku tähelepanu andmete publitseerimisele. Üheks põhjuseks on keerukus, mis kaasneb FAIR printsiipidega kooskõlas olevate andmete publitseerimisega. Kogu teaduskogukonna eesmärk peab olema publitseeritud andmete kvaliteedi tõstmine, et need oleks võimalikult lihtsasti leitavad ja taaskasutatavad. Eesmärgi

saavutamisele aitab eelkõige teadlaste valmidus minna kaasa FAIR printsiipide jälgimisega [4].

2.2 FAIR juhtpõhimõtted

FAIR (leitavus, kättesaadavus, koostalitlusvõime ja korduvkasutatavus) põhimõtted on omavahel seotud, kuid sõltumatud ja eraldatavad. FAIR kirjeldab lühikesi, valdkonnast sõltumatuid põhimõtteid, mida saab rakendada paljude teadlaste väljundite jaoks. Põhimõtted määratlevad omadusi, mida kaasaegsed andmeressursid, tööriistad, sõnavara ja infrastruktuurid peaksid avaldama, et aidata kolmandatel isikutel neid leida ja taaskasutada. FAIR printsiibid saavad seega olla rakendatavad ka teadusvaldkonna väliselt. FAIR printsiibid on kõrgetasemelised soovitused ning nad kuidagi ei pane piire konkreetse tehnoloogia, standardi ega lahenduse osas. Nad toimivad publitseerijatele ja andmetehoidjatele juhendina, et aidata hinnata, kas nende digitaalse teadustöö andmed on leitavad, juurdepääsetavad, koostalitusvõimelised ja korduvkasutatavad [4].

FAIR printsiibid on alles 2016 aastal publitseeritud kontseptsioon ning nende praktikas rakendamine kogu maailmas on algusjärgus. FAIR kõrgetasemelised põhimõtted on loomult sellised, et neid on võimalik implementeerida väga erinevate tehnoloogiate ja rakendustega. Pole veel välja kujunenud kindlaid standardeid. Vaatamata sellele on teadusandmete avaldamise ökosüteemis juba praegu palju andmehoidlaid, kus rakendatakse FAIR põhimõtete erinevaid aspekte [4]. Peatükis 2.3 on selliste teadusandmebaaside ülevaade.

2.2.1 *Findable* - leitavus

Andmete taaskasutamise esimene samm on nende leidmine. Metaandmeid ja andmeid peavad olema lihtsalt leitavad nii inimeste kui ka arvutite jaoks. Masinloetavad metaandmed on andmekogumite ja teenuste automaatseks avastamiseks hädavajalikud [6].

- (Meta)andmetele määratakse globaalselt ainulaadne ja püsiv identifikaator;
- andmeid kirjeldatakse rikkalike metaandmetega;
- metaandmed sisaldavad selgesõnalist nende kirjeldatud andmete identifikaatorit;
- (meta)andmed registreeritakse või indekseeritakse otsitavas ressursis.

2.2.2 *Accessible* - kättesaadavus

Peale andmete leidmist peab kasutaja teadma, kuidas neile juurde pääseda. Selle alla kuulub ka autentimine ja autoriseerimine [6].

- (Meta)andmed saab nende identifikaatori abil kätte standardiseeritud sideprotokolli abil;
- protokoll on avatud, tasuta ja universaalselt rakendatav;
- protokoll võimaldab vajaduse korral autentimist ja autoriseerimist;

- metaandmetele on juurdepääs isegi siis, kui andmed pole enam saadaval.

2.2.3 *Interoperable* - koostalitlusvõime

Andmed tuleb tavaliselt integreerida teiste andmetega. Lisaks peavad andmed analüüsimiseks, salvestamiseks ja töötlemiseks olema lihtsasti liidestavad teiste andmetega [6].

- (Meta)andmed kasutavad teadmiste esitamiseks ametlikku, juurdepääsetavat, jagatud ja laialdaselt kasutatavat keelt;
- (meta)andmete kasutamisel kasutatakse FAIR põhimõtteid järgivat sõnavara;
- (meta)andmed sisaldavad kvalifitseeritud viiteid teistele (meta) andmetele.

2.2.4 *Reusable* - korduvkasutatavus

FAIR lõppeesmärk on andmete taaskasutuse optimeerimine. Selle saavutamiseks peaksid metaandmed ja andmed olema hästi kirjeldatud, et neid saaks erinevates seadetes taaskasutada [6].

- (meta)andmed on rikkalikult kirjeldatud paljude täpsete ja asjakohaste atribuutidega
- (meta)andmed väljastatakse selge ja juurdepääsetava andmekasutuslitsentsiga
- (meta)andmed on seotud üksikasjaliku lähtekohaga
- (meta)andmed vastavad domeeniga seotud ühenduse standarditele

2.3 Teadusandmebaaside ülevaade

Paljud andmehoidlad rakendavad erinevaid FAIR printsiipe. Tabel 1 annab ülevaate sellistest teadusandmebaasidest, mis töötavad kooskõlas FAIR printsiipidega ning võtavad vastu bioloogia valdkonna andmeid. Veerg “Andmebaas” näitab vaadeldavat teadusandmebaasi. Veerg “Formaadid” loendab teadusandmebaasi poolt eelistatavaid andmeformaate. Tabelis 1 on näha, et enamus kirjeldatud andmebaasidest aktsepteerivad publitseerimiseks kõiki formaate. Selle lähenemise miinuseks on see, et publitseerija vastutab ise sobiva andmeformaadi valiku eest, Peatükis 2.2.3 kirjeldatud koostalitusvõime tagamiseks. Peatükis 2.4 vaatleme semantilise veebi *Resource Description Framework* edaspidi RDF andmemudeli serialiseerimise formaate. RDF mudeli kasutamine andmete publitseerimisel võimaldab tagada andmestike koostalitusvõimet. Veerg “FAIR” annab hinnangu teadusandmebaasi vastavust FAIR printsiipidele. Veerg “Kommentaar” lühikese ülevaate teadusandmebaasi kasutamise taustast ja otstarbest [4], [14].

Tabel 1. Teadusandmebaaside ülevaade

Andmebaas	Formaadid	FAIR	Kommentaar
Figshare	kõik formaadid	osaliselt	Andmebaasis on leitavad RDF serialiseerimise formaadis andmefailid. Andmebaasis on leitavad publikatsioonid bioloogiateadusest [13]
Datadryad	Teksti failide jaoks on eelistatud CSV, XML, JSON ja piltide jaoks PDF, JPEG, PNG, TIFF, SVG	osaliselt	Dryad on eelretsenseeritud teadus- ja meditsiinikirjanduse aluseks olevate andmete rahvusvaheline hoidla, eriti selliste andmete kohta, mille kohta pole spetsiaalset hoidlat. Sisu peetakse avaldatud teadustöö lahutamatuks osaks. Kogu Dryad-i materjal on seotud teadusliku väljaandega [12]
Zenodo	kõik formaadid	Jah	Võimalik leida ka OWL kujul olevad andmeid, kuid võrreldes figshare-ga on neid palju vähem. Enamus otsingute puhul ei ole võimalik isegi valida faili formaadiks RDF serialisatsioone, kuna seda on võrreldes teiste otsingutega väga palju [11].
BioStudies	kõik formaadid	Jah	Andmebaas on uus EMBL-EBI ressurss, mis sisaldab bioloogiliste uuringute kirjeldusi, linke teiste andmebaasidesse toetavate andmetele ja arhiivide andmefaile, mis ei mahu olemasolevatesse avalikesse struktureeritud arhiividesse [10].

Biomodels	SBML, CellML, matlab, mathematica	FAIR	Repositoorium on süsteeme kirjeldavate matemaatiliste mudelite jaoks. Mudelite puhul pole oluline, mis modelleerimisvormingut on kasutatud või modelleerimisviisi (ODE, FBA, loogiline, agendipõhine jne) on kasutatud bioloogilise mehhanismi kirjeldamiseks. BioModels aktsepteerib, mis tahes vormingus mudeleid, kuid kureerimise tasandil pakutakse ainult minimaalset tuge SBML-välistele mudelitele. SBML-is esitatud mudelite jaoks viiakse läbi ulatuslik mudelite kontrollimise protseduur (kureerimine) [9].
Fairdom Hub	SBML, CellML ja teised üldlevinud süsteembioloogias levinud formaadid	Jah	FAIRDOM-SEEK on platvorm andmekogumite, mudelite või simulatsioonide, protsesside ja uurimistulemuste jagamiseks. SEEK raamistikus on mitmeid tööristu, mis hõlbustavad FAIR printsiipidele vastavate andmete loomist [8].
Dataverse Harvard	kõik formaadid	Jah	Dataverse on rakendus uurimisandmete jagamiseks, säilitamiseks, tsiteerimiseks, uurimiseks ja analüüsimiseks. See hõlbustab andmete kättesaadavaks tegemist teistele ja võimaldab teiste töid hõlpsamalt korrata. Teadlased, ajakirjad, andmete autorid, kirjastajad, andmete levitajad ja sidusasutused saavad kõik akadeemilise hinde ja veebi nähtavuse [7].

2.4 RDF

RDF (*Resource Description Framework*) on raamistik andmete esitamiseks veebis. See tagab koostalitusvõime rakenduste vahel, mis vahetavad veebis masinale arusaadavat teavet. RDF rõhutab võimalusi, mis võimaldavad veebiressursside automatiseeritud töötlemist. RDF-il on abstraktne süntaks, mis kirjeldab graafipõhist andmemudelit. Elemendid võivad olla IRI-d (*Internationalized Resource Identifier*), tühjad sõlmed või andmetüübiga väärtused. See võimaldab seda kasutada erinevates kasutusvaldkondades. Antud lõputöö seisukohalt on kõige olulisem RDF omadus hõlbustada teadmiste jagamist ja vahetamist tänu sellele, et erinevaid RDF andmestike on lihtne korduvkasutada ja omavahel integreerida [19], [22].

2.4.1 Viited andmeallikatele

RDF raamistiku paremaks arusaamiseks on vaja teada, kuidas toimuvad viitamised andmeallikatele. RDF kasutab ressursside viitamiseks *Internationalized Resource Identifier* edaspidi IRI-sid. IRI on *Uniform Resource Identifier* edaspidi URI laiendus, mis oluliselt laiendab lubatud tähemärkide kogumit. IRI on ainulaadne märkidest koosnev rida. Absoluutselt iga IRI tähistab midagi maailmas. Neid asju millele viidatakse nimetatakse ressurssideks. Näiteks IRI, mis viitab ressursile:

```
<http://purl.obolibrary.org/obo/BTO_0000042>
```

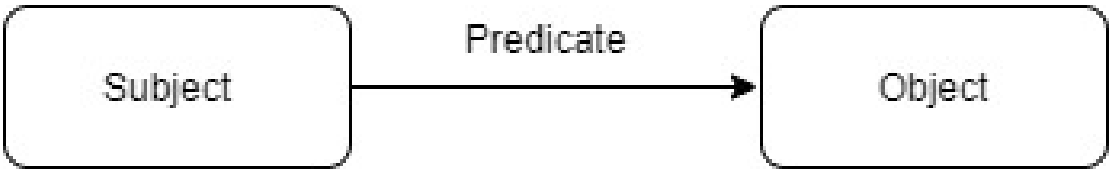
Lisaks ressurssidele viitamisele tähistatakse RDF-s IRI-ga otseseid väärtusi, mida nimetatakse literalideks. Literalidel on andmetüübid, mis määratlevad võimaliku väärtuste vahemiku, näiteks stringid, numbrid ja kuupäevad. Literal RDF-is koosneb mitmest elemendist,

```
"1"^^<http://www.w3.org/2001/XMLSchema\#integer>
```

kus “1” on väärtus ja <http://www.w3.org/2001/XMLSchema#integer> on andmetüübi IRI [23], [24], [21].

2.4.2 Graaf

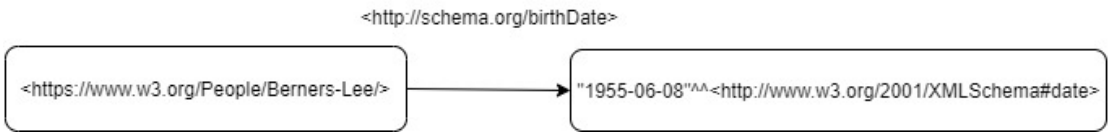
RDF Abstraktse süntaksi põhistruktuuriks on kolmikute kogum. Iga kolmik esitab sõlmede vahelist seost. Iga kolmik koosneb subjektist, objektist ja predikaadist, mis tähistab eelnevate vahelist seost. Selliste kolmikute komplekti nimetatakse RDF-graafiks. RDF graafi saab visualiseerida sõlme ja suunatud kaardiagrammina, milles iga kolmikut on kujutatud sõlme-kaare-sõlme lingina. Joonis 1 illustreerib sellist kolmikut, kus subjekt on seotud objektiga predikaadi abil [21]. Kaare noole suund on oluline ning see on alati suunatud objekti poole. Sõlmed on alati kolmiku subjektid või objektid ning kaarega tähistatakse predikaate [22], [21].



Joonis 1. RDF graaf, millel on kaks sõlme (subjekt ja objekt) ning neid ühendav kaar (predikaat)

2.4.3 RDF serialiseerimise formaatide ülevaade

RDF-il puudub üks konkreetne serialiseerimise formaat. RDF kolmikuid saab esitada mitmel viisil. Järgnevalt vaatame populaarsemaid formaate, nende eeliseid, probleeme ja koodi näiteid. Erinevaid serialiseerimise formaate demonstreerime Joonis 2 näitel. Joonisel 2 välja toodu näidis kolmik annab meile info Tim Berner-Lee sünnikuupäevast. Avades subjekti IRI veebis leiame Tim Berners-Lee biograafia. Tänu sellele on meil võimalik aru saada reaalselt, kelle kohta on antud info. Ilma selleta ei oleks meil piisavalt tausta, et aru saada, kes täpselt on Tim Berners-Lee nime all [20].



Joonis 2. RDF graaf, mille subjektiks on sõlm, mille IRI viitab Tim Berners-Lee biograafiale. Predikaadiks on sünnikuupäev ja literaliks kuupäeva formaadis Tim Berners-Lee sünnikuupäev. Selle kolmikuga anname edasi järgneva sisu: “Tim Berners-Lee on sündinud 1955.06.08.”

RDF/XML RDF/XML formaat on esimene ja tõenäoliselt kõige kuulsam RDF serialiseerimise formaat. Selle juures on huvitav märkida, et see on spetsialistide seas vihatud. Selle põhjuseks on antud formaadi leiutamise aeg. Seda võeti esimest korda kasutusel ligikaudu 20 aastat tagasi, siis kui enamus süsteeme oskas töödelda XML formaati. Seega tundus loogiline kirjutada RDF-i XML keeles. Kahjuks osutus, aga et RDF ja XML omavad kahte erinevad põhimõttelist kontseptsiooni: kolmikute graaf ja puu struktuuriga dokument. See muudab RDF/XML kontseptuaalselt keeruliseks võrreldes teiste standarditega. XML-i arendaja jaoks on süntaks tuttav, kuid isegi tema jaoks ei tule RDF/XML-ist selgelt välja kolmikute struktuur, mis teeb RDF/XML-iga töötamise keeruliseks. Tegemist ei ole kasutajasõbraliku formaadiga [20], [19], [25].

```
<?xml version="1.0"?>

<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:schema="http://schema.org/">
```

```
<rdf:Description rdf:about="https://www.w3.org/People/Berners-Lee/">
  <schema:birthDate>1966-06-08</schema:birthDate>
</rdf:Description>

</rdf:RDF>
```

RDFa RDFa on RDF, mis on pakendatud HTML-i sisse. HTML elementide atribuutide lisamisega on võimalik hõlpsasti lisada oma veebilehtedele semantilist konteksti. Google on võimeline seda töötlema, et muuta otsingu eelvaateid paremaks, kuigi nad ise soovitavad eelistada selleks JSON-LD formaati. RDFa kasutuselevõtt ei ole kunagi olnud väga laialdane, kuna vajaliku koodi kirjutamine muutis HTML-i mitte loetavaks ja isegi Google varem ei osanud alati seda lugeda korrektselt.

RDFa erineb põhimõtteliselt teistest vormingutest, kuna see ühendab RDF-i vaateandmetega. See tähendab, et see muudab kirjutatud HTML dokumendid suuremaks ja keerukamaks ning selle parsimine kolmikuteks on ressurssinõudvam kui näiteks N-kolmikute formaadis olevaid andmeid. See muudab RDFa vähem kasulikuks, kui teie rakendus tugineb paljudele RDF andmetele. RDFa on mõistlik kasutada, kui soovite olemasolevale veebilehele lisada piiratud kogu-se metaandmeid [20], [19].

```
<div about="https://www.w3.org/People/Berners-Lee/">
<p>
  Tim is born on
  <span property="http://schema.org/birthDate">1955-06-07</span>
</p>
</div>
```

Notation3 (N3) Eesmärk oli luua midagi paremat kui RDF/XML, mille tulemusena tuli välja N3. Erinevalt RDF/XML-ist sarnaneb N3 RDF subjekt-predikaat-objekt mudeliga. See muudab N3 loetavamask ja palju lihtsamaks ning võimaldab kasutajal aru saada, kuidas RDF töötab. Kasutades N3-s eesliiteid võib väljud olla üsna kompaktne. Siiski on N3-l mitmeid probleeme. N3 serialiseerimine on masinale kulukas, mis mõjutab rakenduste jõudlust. Eelistatud on kasutada N3-s välja arenenud Turtle standardit, mille parsimine on vähem kulukas [20], [19], [26].

```
@prefix tim: <https://www.w3.org/People/Berners-Lee/>.
@prefix schema: <http://schema.org/>.
```

```
<tim> schema:birthDate
"1955-06-08"^^<http://www.w3.org/2001/XMLSchema#date>.
```

Turtle Turtle ehk *Terse RDF Triple Language* on välja arenenud N3-st. Turtle on lihtsustatud versioon N3 formaadist. Sellesse on jäetud ainult RDF jaoks vajalikud elemendid. Tänu sellele on Turtle parsimine vähem ressursikulukas võrreldes N3-ga, kuid see on endiselt keerukam võrreldes N-Triples formaadiga. Süntaktiliselt on standard väga sarnane N3-ga. Meie näite puhul väljund on identne [20], [19], [27].

```
@prefix tim: <https://www.w3.org/People/Berners-Lee/>.
@prefix schema: <http://schema.org/>.

<tim> schema:birthDate
"1955-06-08"^^<http://www.w3.org/2001/XMLSchema#date>.
```

N-Triples N-Triples on Turtle väga lihtne alamhulk, mis omakorda on N3 lihtne alamhulk. N-Triples sees on veel vähem keerulist funktsionaalsust kui Turtle formaadis. N-Triples ei toeta ka eesliiteid ega mingeid väljamõeldud funktsioone. See muudab N-Triplesi serialiseerimist triviaalseks. Seetõttu on veebis kätte saadaval palju N-Triple-i teeke ning seda on samuti kerge ise kirjutada. See muudab ka serialiseerimise ja parsimise väga efektiivseks. Eesliidete ja funktsioonide kasutamine muudab aga väljundit väga pikaks ja natukene keeruliseks lugejale. Lisaks pikad IRI-d viivad selline, et tõenäoliselt tekkib vajadus kasutada faili pakendamist selleks, et mitte raisata salvestusmahtu [20], [19].

```
<https://www.w3.org/People/Berners-Lee/>
<http://schema.org/birthDate>
"1955-06-08"^^<http://www.w3.org/2001/XMLSchema#date>.
```

JSON-LD JSON on kahtlemata kõige populaarsem viis veebirakendustes andmete järjestamiseks. JSON-LD on JSON-i laiendus. Seega on JSON-LD kehtiv JSON. Kasutajal on võimalik muuta tavaline JSON RDF-ks, lisades @context annotatsiooni objekti. See objekt toimib peamiselt kaardistamisena. Selle abil muudetakse tavalised JSON võtmed väljamõeldud linkideks RDF-klassidele ja omadustele. Konteksti saab lisada, kas lisades oma HTTP-vastusesse @content *Header*-i, lisades lingi oma JSON-i kehasse või lisades kogu @context-objekti JSON-sse.

JSON-LD on lihtsasti loetav ja kindlasti tuttav enamus kasutajatele. Kahjuks on JSON-LD parsimine keeruline ja kulukas, kui eesmärk on kasutada JSON objekti asemel RDF mudelit. Seega ei ole soovitatav kasutada JSON-LD formaati mitte-triviaalsete API-de puhul [20], [19].

```
{
  "@context": {
    "schema": "http://schema.org/"
  },
  "@id": "https://www.w3.org/People/Berners-Lee/",
  "schema:birthDate": "1955-06-08"
}
```

2.5 Andmete ontoloogia

Ontoloogia on andmete mudel, mis representeerib teadmisi kui mõistete kogumit domeenis ja nende mõistete vahelisi seoseid. Ontoloogia võimaldab programmeerijal avatud, tähendusrikkal viisil täpsustada mõisteid ja seoseid, mis ühiselt iseloomustavad mõnda huvipakkuvat valdkonda. Ontoloogia eeliseks on selgesõnaline esmaklassiline kirjeldus. Seega kui see on välja töötatud ühel eesmärgil, saab seda avaldada ja uuesti kasutada muudel eesmärkidel. Ontoloogia kirjutamiseks on palju viise ja erinevaid arvamusi selle kohta, millised definitsioonid ühes ontoloogias peaksid olema. Praktikas on ontoloogia sisu suuresti sõltuv rakenduste tüübist, milles seda kasutatakse [36].

2.5.1 Ontoloogia lisamise ja koostamise alused

Ontoloogia lisamisel tuleb olla väga ettevaatlik ja jälgida seda, et ontoloogia ei oleks lisatud ainult sellepärast, et see oleks olemas muidu muutuvad andmed liiga küllastunuks. Lisaks peab seoste lisamisel võimalikult palju kasutama juba olemasolevaid skeemasid ja mõisteid. Uusi nimetusi tuleb lisada vaid väga suurel vajadusel ning alati otsida juba loodud variante, kuna need on juba teatud määral muutunud standardiks. Näiteks pole mõistlik luua uut mõistet “sex”, kuna on kindel et mõistet “sugu” on juba varasemalt defineerinud paljud ontoloogiad ning tuleb vaid leida enda ontoloogia jaoks sobiv termin teisest ontoloogiast [37].

2.5.2 Ontoloogia mõistete nimetamine

Igapäevaelus kasutavad inimesed asjade jaoks tihti napisõnalisi, mitmetähenduslikke ja krüptilisi nimesid. Selleks on palju põhjuseid. Meie igapäevases keeles saame aja ja vaeva säästmiseks kasutada lühinimesid ja lühendeid, lootes et mõiste ei jää ebaselgeks täna ümbritsevale kontekstile. Kahjuks võib kontekst ajas muutuda. Tänapäeval on salvestusruum odav ja hea tarkvara suudab väärtuseid automaatselt kuvada lühikeste nimedena, säilitades samal ajal pikemad

ja üheselt mõistetavad terminid. See võimaldab meil mitte muretseda infole viitamise pikkuse pärast [37]. Peatükis 2.4.1 kirjeldasime, kuidas on võimalik viidata andmeallikatele. Kuid miks on see hea viis?

Kasutades IRI-sid võime anda globaalselt ainulaadset nime kõigele, millest rääkida tahame. Saame:

- anda oma uuringule globaalse ainulaadse nime;
- nimetada iga uuringu teema;
- nimetada iga oma andmetabeli rea;
- nimetada seoseid, mida meie veerud väljendavad;
- anda nime väärtustele, mida taaskasutame.

Muidugi ei pea kõigele nime määrama, kuid selline võimalus on meil olemas.

2.5.3 Pikkade nimede lühendamine

IRI-d võivad olla pikad ja sarnaste IRI-de loendid võivad olla üleliigsed. Nende lühendamiseks on kasutusel, teave kaotamata, CURIE *Compact URI* meetod. CURIE koosneb eesliitest ja sufiksist, kus eesliide tähistab pikemat baas-IRI-d. Eesliidet laiendades ja järelliidet lisades jõuame tagasi täieliku IRI juurde.

Näiteks kui omistame OBO prefiksile IRI-ga http://purl.obolibrary.org/obo/ sufiksi, siis saame CURIE obo OBI_0000070 laiendada aadressile http://purl.obolibrary.org/obo/OBI_0000070 ilma igasuguse ebaselguseta. Kõik dokumendid, mis sisaldavad CURIE-sid, peaksid määratlema ka eesliited. Tabelis 2 on välja toodud RDF ökosüsteemis üldkasutatavad eesliited [37], [22].

Tabel 2. Üldlevinud eesliited

Eesliide	IRI	Nimi
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	Resource Description Framework
rdfs	http:/www.w3.org/2000/01/rdf-schema#	RDF skeem 1.1
xsd	http://www.w3.org/2001/XMLSchema#	XML skeem
owl	http://www.w3.org/2002/07/owl#	Web Ontology Language
obo	http://purl.obolibrary.org/obo/	Open Biological and Biomedical Ontologies

Ühele mõistele üks nimi IRI-d võimaldavad meil määrata nii palju nimesid kui soovime. Peab meeles pidama, et nimedest on kõige rohkem kasu, kui kõigel, millest rääkida tahame, on ainult üks nimi. Kui kasutame sama asja jaoks erinevaid nimesid, siis on vaja teha lisatööd, et aru saada, kas mõtleme ühte või teist. Teadusvaldkonnas toimuvad mõistete ja nimede muutused pidevalt, vastavalt uutele avastustele ja toimunud muutustele valdkonnas [37].

2.6 Relatsioonilise andmebaasi vastavusse seadmine RDF-ga

Relatsiooniliste andmete vastandamisel RDF-ks on kaks pealmist lähenemist:

1. Andmete otsene kaardistus (*Direct Mapping*).
2. Andmete kaardistamine RDB-RDF kaardistuskeele R2RML abil (*R2RML Mapping language*).

Otsene kaardistamine kaardistab relatsioonilise andmebaasi RDF graafiks, ilma et oleks võimalik kontrollida genereeritud graafi kuju. See lähenemine nõuab hilisemat kasutaja poolset sekkumist ning saadud RDF teisendamist. Samas R2RML kaardistamiskeel võimaldab eelnevalt defineerida kõik vajalikud seosed, mis võimaldab genereerida lõpliku RDF graafi. R2RML on populaarseim ja peamine kasutusel olev lähenemine, mis kirjeldab kohandatud kaardistamisreegleid. R2RML keeles defineeritud vastavustega on võimalik muuta relatsioonilised andmed RDF andmekogumiteks. [35] [34].

2.6.1 R2RML kaardistuse loomine

R2RML kaardistuste kirjutamist on võimalik teha tavalise tekstiredaktoriga. R2RML kaardistuse loomine ei erine selles mõttes kuidagi programmeerimiskoodi kirjutamist. R2RML kaardistuse loomine nullist on keeruline protsess ja nõuab sügavat keele tundmist ning selle pärast on olemas mitmeid tarkvarasid, mis lihtsustavad kaardistuste kirjutamist. Enamlevinud tarkvarade ülevaade:

Ontop Plugin Protege jaoks *Plugin*-i abil saab eksportida ja importida R2RML faile. Üldiselt on selle tööriista kasutajad tehnikud, formaalse ja eriti kirjeldusloogika eksperdid. *Ontop* on keeruline kasutamiseks tavalise domeeni eksperdi jaoks ning see nõuab korralike R2RML keele, SQL ja Ontoloogia teadmisi [50], [32].

Karma Systems *Karma* on kaardiredaktor, mis pakub graafilist kasutajaliidest vastavuste visualiseerimiseks ja redigeerimiseks. Veelgi enam, Karma saab automaatselt soovitada kaardistusi ja toetab R2RML-i. Vastendused RDBMS ja RDF vahel kuvatakse ontoloogia jaoks puukujulise struktuurina ja andmebaasi skeemi tabelite paigutuste abil. Tööriista piirang seisneb

puu paigutuse puudulikus infos visualiseerimisel. Selline paigutus ei suuda näidata andmebaasi skeemi, ontoloogia ja kaardistamise täielikku struktuuri piisavas mahus. Osa infot läheb sellise visualiseerimise juures kaduma [44].

Map On *Map On* on ontoloogilise kaardistuse graafiline keskkond, mis aitab erinevatel kasutajatel - domeeniekspertidel, andmete omanikel ja ontoloogiainseneridel - luua ja säilitada kaardistusi andmebaasi ja domeeni ontoloogia vahel, kasutades R2RML keelt. Ontoloogia kaardistamiskeskkond pakub graafikute paigutusel põhinevaid vastenduste visualiseerimist ning toetab IRI mustrite ja SQL päringute automatiseeritud genereerimist R2RML-lausete jaoks. Selle miinuseks on see, et R2RML kaardistuse loomiseks on vaja ontoloogiat, mis juba vastab kasutaja andmetele, mida üldjuhul kasutajal ei ole. Samuti ei lihtsusta see, kuidagi ontoloogiliste mõistete lihtsustamist ning andmebaasi alamhulga välja võtmiseks on endiselt vaja domeeni eksperdil korralikku SQL teadmist [43].

2.7 Teadusandmete publitseerimisprotsessi ülevaade

Süsteembioloogia laboris on kasutusel erinevatest katsetest pärinevate andmete linkimiseks üks andmebaas. Andmebaas on erinevate väiksemate rakenduste kasutuses, mis töötavad ainult andmebaasi osaga. Peale uuringu teostamist on uuringutulemused koondatud ühte andmebaasi. Teadlase eesmärk on avaldada ühte konkreetset uuringut kirjeldavaid andmekogumeid. Soovitud andmed peavad olema seotud ainult konkreetse uuringuga. Andmete väljavõtmisel on oluline saada vaid katse jaoks relevantsed andmed. Järgmise sammuna on vaja kaardistada andmete ja ontoloogia terminite vahelisi seoseid. Näiteks tabeli veerul “sex” võib olla ontoloogia termini *phenotypic sex* http://purl.obolibrary.org/obo/PATO_0001894 vaste. Peatükis 2.7.1 kirjeldame ideaalset protsessi, mida teadlane peaks tegema, et kooskõlastada oma andmed FAIR põhimõtetega viies relatsioonilised andmed RDF kujule ja lisades neile annotatsioone.

2.7.1 Andmete käsitsi ettevalmistus publitseerimiseks

Enne andmete publitseerimist on Süsteembioloogia labori teadlasel vaja luua oma andmemudelile vastavat ontoloogiat, vastandada relatsioonilises andmebaasis olevad andmed ontoloogiaga ja eksportida andmed relatsioonilisest baasist RDF kujule. Käsitsi protsessi kirjelduses kasutame ainult juba olemasolevat avatud lähtekoodiga tarkvara.

Andmemudeli ontoloogia loomine Enne ontoloogia loomist tuleb kaardistada andmete ja ontoloogia terminite vahelisi seoseid. Eelistatud on luua nimekiri enda andmetest ja neile vastavatest ontoloogia terminitest. Terminite mugavamaks otsimiseks on mitmeid tööriistu nagu *Ontobee* ja *BioOntology*. Ontoloogiatermini kasutamine tähendab selle IRI kasutamist, kuid ideaalis

ainult sellest ei piisa. Me soovime säilitada ka selle sildi, definitsiooni ja muid omadusi. Selle saavutamiseks tugineme MIREOT põhimõtetel. MIREOT põhimõtted määravad minimaalse informatsiooni koguse, mida peab jätma viidates ontoloogiale. MIREOT nõuetega kooskõlas töötab *Ontofox* rakendus, mis võimaldab kaardistatud terminid pakendada uude ontoloogiasse. Kui tekib probleem sobivate terminite leidmisega, siis on võimalik hiljem lisada puuduolev info Protege ontoloogia manipuleerimise tarkvaraga [28], [29], [31], [30].

Relatsioonilise andmemudeli vastavusse seadmine ontoloogiaga Meil on vaja baasist välja võtta relevantsete andmete alamhulk. Uuringu tulemuste eksportimisel andmebaasist ei piisa lihtsalt tabelite mingi osa väljavõtmisest, kuna peavad olema säilitatud andmetevahelised seosed. Üheks selleks sobivaks tööriistaks on *Jailer*, mis on mõeldud andmebaasi alamhulga, skeemi info väljavõtmiseks. Kasutades *Jailer*-i sisse ehitatud *Data Browser* või *Extraction Model* töövahendeid on võimalik kirjeldada relatsioonilises andmebaasis olevaid andmeid ja nende seoseid, mida soovime andmebaasist välja võtta. Peale väljatõmbamismudeli defineerimist on seda võimalik taaskasutada andmebaasist andmete kiireks ja hõlpsaks väljavõtmiseks.

Andmete annoteerimise keskseks tarkvaraks on meil *Protege*, kuna see võimaldab ontoloogiat muuta ja vastandada meie andmetega. Selleks, et viia relatsioonilised andmed RDF kujule peame looma kaardistuse RDB ja RDF vahel. Kaardistuse tegemisel kasutame Peatükis 2.6 kirjeldatud R2RML kaardistust. Kaardistuse teeme vastu eelmises peatükis loodud ontoloogiat. RDB ja RDF vahelise kaardistuse tegemist aitab meil teha *Protege* plugin *Ontop*. *Plugin*-i abil automaatselt genereerime kaardistuse, mida edasi saame muuta vastavalt meie nõuetele. Peale kaardistuse valmimist vastandame relatsioonilised andmed ontoloogiaga jooksutades *Ontop Materialize Triples* käsku. Edasi on meil võimalik salvestada andmeid meid huvitavasse RDF serialiseerimise formaati või säilitades ontoloogiat OWL kujul [50].

Võimalik töövoog andmete alamhulga eksportimiseks ja ontoloogia vastandamiseks:

- loome väljavõtte mudelit sõltuvalt uuringutulemuste vajadusest;
- ekspordime *Jailer*-i abil andmebaasi alamhulga SQL lausete kujul;
- loome uue ajutise andmebaasi koos skeemaga, mis katab meie eksporditavate andmete vajaduse;
- impordime *Jailer*-i abil genereeritud *INSERT* laused ajutisse andmebaasi;
- avame varasemalt *OntoFox* abil loodud ontoloogia *Protege*-s;
- protege *Ontop* plugini abil loome ühenduse meie ajutise andmebaasiga;
- impordime *Ontop* pluginisse RDB ja RDF vahelise vastavuste faili;
- jooksutades *Materialize Triples* käsku sisetab *Ontop*, kasutades eelmises sammus sisse toodud vastavusi, andmed ontoloogiasse;

- salvestame andmetega täidetud ontoloogia meid huvitavasse formaati.

2.8 Ülevaade ontoloogia ja RDF tööriistadest

Veebis on palju erinevaid lahendusi, mis võimaldavad transformeerida relatsioonilise andmebaasi RDF-ks. Analüüsi käigus selgus, et paljud teegid ei ole enam arenduses, põhinevad vanadel tehnoloogiatel või omavad kasutuspiiranguid. Järgnevalt on välja toodud analüüsitud ja katsetatud rakendused.

PyRDB2RDF Teeki pole väga ammu arendatud ning põhineb Python 2.7 tehnoloogial. Rakendus suudab teha relatsioonilise andmebaasi transformeerimist RDF-ks kasutades *Direct Mapping*-ut. Tegemist vana implementatsiooniga, mis kasutab enda sees populaarset RDFLib ja sqlAlchemy teeki. Lisaks on rakenduse puuduseks suur tundlikus andmebaasi struktuuri osas, kui näiteks andmetabelis on kehvasti defineeritud *foreign key* siis läheb RDF genereerimine katki. Lihtsamate test andmete puhul on tulemuseks valiidne RDF. Suurema reaalsete andmetega baasis puhul ei õnnestunud saade seda tööle. Rakendust oleks vaja kasutusele võtmiseks kindlasti ümber kirjutada [38].

Stardog *Stardog* ei ole avatud lähtekoodiga, kuid on kättesaadav tasuta kogukonna litsents piiratud funktsionaalsusega. Stardog kasutab relatsiooniliste andmete transformeerimiseks RDF-i R2RML kaardistamist. Stardog on kasutajasõbralik töövahend, mis omab korraliku kasutusjuhendite raamatukogu. Selle miinusteks on avatud lähtekoodi ja tasuta versiooni puudumine ja vajadus teada R2RML keelt RDB-RDF kaardistuste tegemiseks [39].

Jena Framework Java peale ehitatud *Jena* raamistik omab endas erinevaid mooduleid, mis võimaldavad *Jena* kasutajal teha tööd RDF-ga, ontoloogiatega ja SPARQL-ga. SPARQL on päringute keel, mis võimaldab pärida ja manipuleerida RDF-i. Tegemist on SQL alternatiiviga RDF kolmikute maailmas. Lisaks võimaldab *Jena* raamistik talletada andmeid kolmikutena raamistiku poolt toetatud TDB *store*-s. Samuti võimaldab teha tööd RDFS ja OWL formaatidega. Kahjuks ei ole *Jena* raamistikus sisse ehitatud võimalust liidestuda relatioonilise andmebaasiga. Selle jaoks on vaja luua vahelüli, mis muudab RDB baasis olevad andmed RDF kujule enne seda, kui on võimalik *Jena*-ga nendele ligi saada [40].

R2RML parser Java peale ehitatud *R2RML parser* kasutab relatsiooniliste andmete transformeerimiseks RDF-i R2RML kaardistamist. *R2RML parser* toetab JDBC andmebaase, kuid

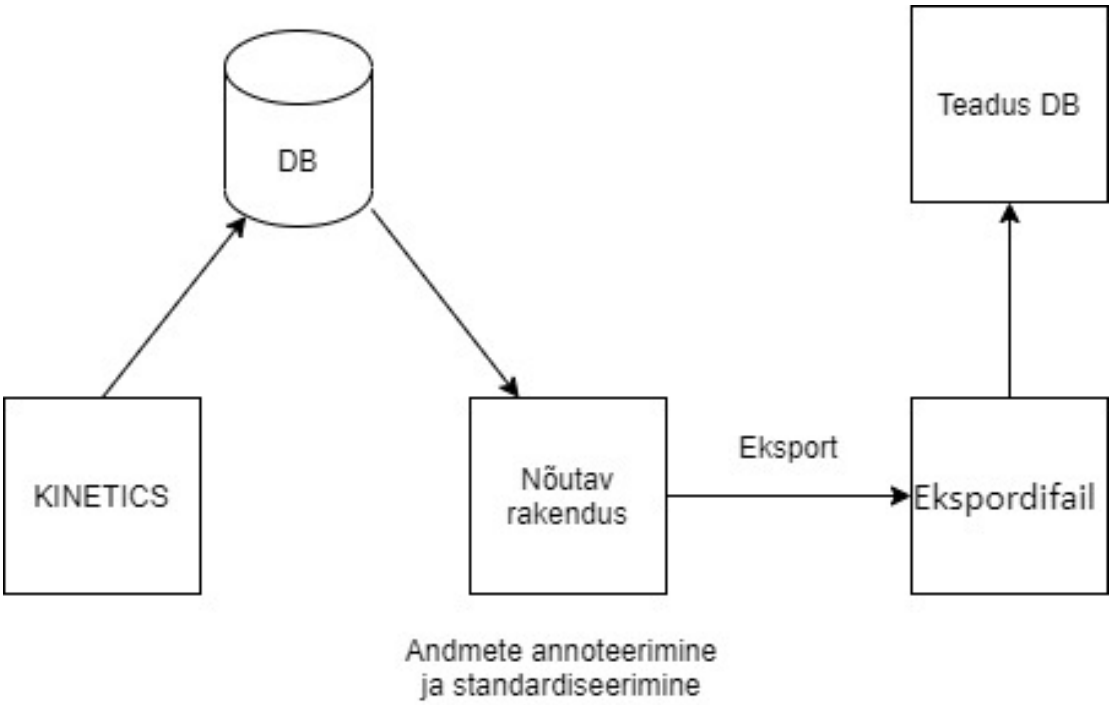
selle arendamine pole olnud aktiivne viimased 5 aastat. Samuti kasutab see vana Java 7 ja *Jena* 2 versiooni [34].

D2R Server *D2R Server* on suhteliste andmebaaside sisu avaldamise vahend. *D2R Server* kasutab relatsioonilise andmebaasi RDF-i transformeerimiseks D2RQ kaardistamiskeelt. D2RQ on sarnaselt R2RML keelega mõeldud andmebaaside kaardistuseks nii, et peale transformeerimist ei oleks vaja RDF-i täiendavalt muuta. D2RQ on tunduvalt vähem populaarne R2RML-st. *D2R Server*-it sai testitud reaalsel *Kinetics*-i test andmebaasil. Rakenduse seadistamine on elementaarne ning seda on võimalik panna nii serverisse kui oma arvutisse. Rakenduse abil õnnestus genereerida automaatne D2RQ kaardistus, kuid edasine RDF loomine ebaõnnestus. Osutus, et *D2R Server* on samuti tundlik andmebaasi skeemi suhtes. Selleks, et seda tööle saada on vaja näha vaeva andmebaasi korrastamisega. Teine variant on osata D2RQ kaardistamiskeelt, et luua kõik vajalikud vastavused käsitsi. Samuti on suureks probleemiks see, et *D2R Server* ei ole hetkel arenduses ja projekt seisab. [41].

Protege *Protege* on avatud lähtekoodiga ontoloogia loomise, muutmise ja haldamise graafiline rakendus. Protege võimaldab importida nii RDF erinevaid standardeid kui OWL-s kirjutatud ontoloogiad. Lisaks on *Protege* jaoks kirjutatud hulgaliselt plugineid, mis laiendavad Protege baasfunktsionaalsust. Peale RDF-i või ontoloogia importimise on *Protege* graafilise liidese abil muuta andmete sisu. Võimalik on andmemudeleid täiendada, kustuda, luua vajalike lisa seoseid ja palju muud. RDF kolmikute manipuuleerimise või ontoloogia täiendamise järel on *Protege*-s võimalik eksporteerida saadud tulemus nii RDF kui OWL kujul. *Protege*-d enda sõnul toetab rakendust akadeemiliste, valitsuse ja ärikasutajate kogukond, kes kasutab Protege teadmistepõhiste lahenduste loomiseks nii erinevates valdkondades nagu biomeditsiin, e-kaubandus ja organisatsioonimudelid. Kui kasutaja eesmärk on teha tööd ontoloogiatega, siis *Protege* on tema esimene valik vabatarkvara maailmast [32].

2.9 Nõuded arendatavale rakendusele

Peatükis 2.7.1 käisime üle sammud, mida on vaja teha, et andmebaasi koondatud andmed oleks võimalik publitseerida RDF kujul. Joonis 3 illustreerib nõutava rakenduse paigutust andmete publitseerimise skoobis. Nõutava rakenduse sisendiks on kasutaja andmebaas ja väljundiks genereeritakse ekspordifail, mida on võimalik publitseerida teadusandmebaasi. Selleks, et rakendus lahendaks relatsioonilisest andmebaasist, FAIR printsiipidega kooskõlas olevate andmete, publitseerimisega kaasnevad probleemid peab rakendus põhifunktsionaalsuselt suutma:



Joonis 3. Nõutava rakenduse positsioon andmete publitseerimise pildis

- eksportida relatsioonilises baasis olevad andmed;
- eksporteeritavatele andmetele lisada annotatsioone;
- tegema RDB-OWL ja RDB-RDF konverteerimist.

Lisaks kohustuslikele põhifunktsionaalsuse nõuetele on Süsteembioloogia labori poolt defineeritud lisa nõuded:

- tegemist peab olema eraldiseisva rakendusega;
- andmete eksport peab töötama ka osaliselt defineeritud ontoloogiaga;
- peab olema võimalik teha andmete vastandamist ontoloogiaga;
- peab saama eksportida andmebaasi alamhulga;
- rakendus peab olema kasutatav domeenispetsialti jaoks, kellel puuduvad sügavad teadmised semantilisest veebist, kaardistuskeeltes ja RDF-st;
- rakendus peab olema lihtsasti paigutatav ülikooli serveritesse, kuna eesmärk on pakkuda rakendust vabatarkvarana;
- rakendus peab lihtsustama ontoloogia mõistete otsimist.

Sellist tööriista, mis võimaldaks teha seda, mida nõutakse ei ole vabavarana olemas. Mitmes lähenemises on tehtud katsetusi automatiseerida kaardistamisprotsessi osa. Kuid, kuna andmebaasi skeemast vajaliku semantika välja võtmine, et luua korrektne kaardistamine relatsioonilise andmete ja ontoloogia vahel on väga keeruline, on käsitsi kaardistamine endiselt peamine töövoog. See on ajakulukas ja nõuab kõrgetasemelist asjatundlikkust [43].

3 Arenduse faas

Peatükis 2.9 said kirjeldatud nõuded täiuslikule rakendusele. Iseloomustame seda täiuslikuna, kuna sellise rakenduse loomine on väga keerukas ja aeganõudev. Seda väidet toetab see, et semantiline veeb on teadusvaldkonnas teemaks ligikaudu 20 aastat. Süsteembioloogia labori teadusandmete publitseerise probleem on globaalne mure, kuid täiusliku lahendust pole endiselt olemas. Bakalaureuse töö raames on see võimatu eelkõige piiratud aja tõttu.

Süsteembioloogia laboriga sai otsustatud, et proovime luua rakenduse prototüübi, mis suudab:

- luua ühenduse Postgresql andmebaasiga,
- loodud ühenduse järgselt on võimalik valida andmebaasi tabelid ja veerud kaardistuste loomisel,
- aitab otsida ontoloogia mõisteid,
- loob kaardistused RDB ja mõistete vahel,
- ekspordib andmed RDF kujul.

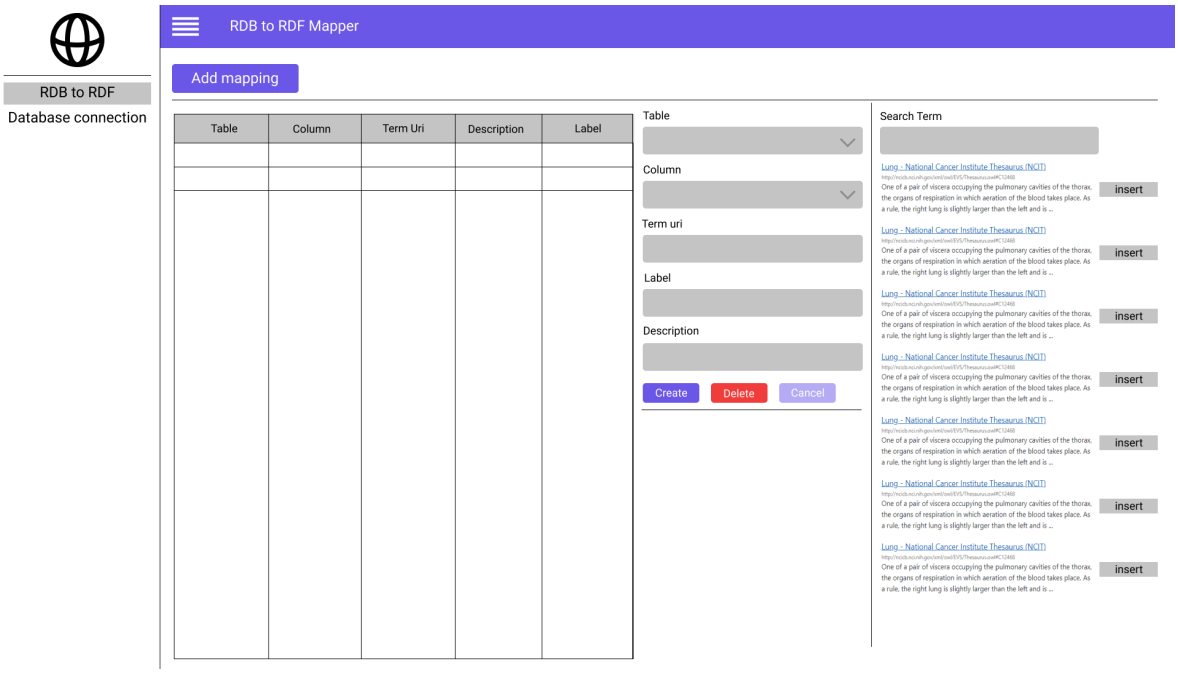
Prototüübi eesmärk on läbi testida hüpoteesi, et kasutades analüüsi käigus leitud teeke on võimalik luua RDB-RDF transformeerimine lisatud annotatsioonidega.

3.1 Kasutajaliidese prototüüp

Rakenduse vaade sai kokkulepitud koostöös Süsteembioloogia laboriga. Algselt sai arendatud lihtne *JavaFX* rakenduse vaade, mida hiljem vahetasime välja veebiliidese vastu. Kasutajaliidese kokkuleppimine käis kasutades iteratiivse kasutajaliidese prototüüpimise abil. Esmase prototüübi aluseks oli *JavaFX*-is loodud vaade, mis sai peaaegu üks ühele üle viidud Figma prototüübi vaatesse. Lõppfaasis toimus iteratsioonile Marko Vendelini poolt tagasiside andmine ja prototüüpi muudatuste sisse viimine. Lõplikult kokkulepitud prototüübi vaade Joonis 4.

3.2 Kasutatud tehnoloogiate valimine

Eelanalüüsi käigus selgus, et rakendusel peab olema kasutajaliides ja taustarakendus. Süsteembioloogia seisukohalt oli eelistatuim variant teha arendust Python-i baasil kirjutatud raamistikus Pyramid, kuna nende tiimis on olemas kompetents selle arendamiseks. Süsteembioloogia labori kahjuks ei olnud võimalik valida Python-is kirjutatud raamistiku, kuna analüüsis selgus, et arenduseks sobivamad teegid on ehitatud Java baasil. See tähendab, et Pythoni valikul tuleks kasutada ümbrist, mis lubaks kasutada Java klasse ja teeke Python-is. On olemas Python teegid, mis võimaldavad seda teha, kuid nende kasutamine lisab rakendusele täiendava keerukuse. Tekib keerulisem seadistamise vajadus, kannatab koodi loetavus ja võimalike probleemide la-



Joonis 4. Pealehe kasutajaliidese prototüüp

hendus on raskem, kuna taolise lähenemise kohta käivat infot on vähem võrreldes tavalise Java implementatsiooniga. Seega sai kindluse mõttes otsustatud valida raamistik, mis töötab Java baasil.

Tabel 3. Vajalike funktsionaalsuste alternatiivsed variandid Java-s ja Python-is

Funktsionaalsus	Java põhine	Python põhine
RDF failidega töö. RDFlib on väiksema funktsionaalsusega	Jena API	RDFlib
Töö R2RML kaardistuskeelega	R2RML-api	puudub
RDB-RDF transformatsioon	*Ontop*	puudub hea variant

Algselt sai alustatud töölaua rakenduse arendamist *JavaFx* baasil, kuid kiirelt sai ümber otsustatud veebiliidese kasuks. Põhjuseks on see, et *JavaFX* rakendus ei pruugi joosta sama hästi igas operatsioonisüsteemis. Lisaks arendada kaasaegset veebiliidest kasutades aktuaalset kasutajaliidese raamistiku on lihtsam ja eelistatum.

Sobiva kasutajaliidese raamistiku tehnoloogia valikul tuginesime järgnevale:

- raamistiku arendamise paindlikkus;
- raamistiku arendamiskiirus uuele kasutajale;
- raamistiku aktuaalsusele.

REST teenuseid serveeriva raamistiku valikul tuginesime järgnevale:

(*fullstack*). See võimaldab arendada täisrakenduse ühe tehnoloogiapinu peale. Arvestades, et Süsteembioloogia laboril puudub kompetents Java-s ja JavaScript-is on *Vaadin* hea raamistik, mille omandamine võtab vähem aega, kuna kahe uue tehnoloogia asemel on vaja õppida üks [46].

Seega välja valituks osutus täisrakenduse-pinu raamistik Vaadin, mis jaguneb kaheks erinevaks kontseptsiooniks: *Flow* ja *Fusion*. *Vaadin Flow* võimaldab kirjutada kogu veebirakenduse kasutades selleks, vaid Java-t. Arendajal ei ole selleks vaja üldse kasutada HTML-i ega JavaScript-i. *Vaadin Fusion* on raamistiku modernsem ja alternatiivne lähenemine, mis võimaldab luua minimaalse konfigureerimisega Typescript kasutajaliidese koos Java taustarakendusega. Taustrakenduse jooksutamiseks kasutab *Vaadin Spring Boot*-i. Kasutajaliidest kirjutatakse *Fusion* lähenemises sarnaselt *React* ja *Vue*-ga deklaratiivsete mallide abil. Lisaks *Vaadin*-iga tulevad kaasa UI (*User Interface*) komponendid, mis omavad rikkalikku funktsionaalsust, mis võtab arendaja tööd vähemaks. Näiteks *Vaadin*-is on automaatselt lahendatud vormide ja andmete sidumine. Prototüübi arendamiseks sai valitud *Vaadin Fusion* lähenemine, kuna see võimaldab kasutajaliidese koodi tükeldada võimalikult väikesteks loogilisteks komponentideks ning hoida kasutajaliidese ja taustrakenduse koodi eraldi [46].

3.2.1 Vaadin Fusion

Vaadin Fusion sai valitud, kuna see võimaldab arendada *full stack* veebirakendust kasutades tavaarendajale tuttavamaid võtteid. Nimelt *Vaadin Flow* valiku miinuseks oleks see, et arendaja jaoks, kes pole kunagi varem kirjutanud UI-d Java-s oleks seda keeruline mõista. *Vaadin Fusion* on disainitud just Java taustrakenduse jaoks. Raamistik genereerib Java lõpp-punktide (*endpoint*) juurdepääsuks automaatselt Typescript asünkroonsed funktsioonid. Funktsioonid genereeritakse automaatselt Java klasside põhjal, mis tähendab, et arendajal on vaja kirjuta vähem koodi. Tänu sellele on taustrakenduse kasutamine *Vaadin Fusion* kasutajaliidesest sama lihtne kui asünkroonse funktsiooni väljakutsumine. Prototüübi arenduseks langes valik täielikult *Vaadin Fusion* peale, mis aga ei takista tulevikus kasutada edasi arenduseks *Vaadin Flow* põhimõtteid. *Vaadin* võimaldab kasutada *Flow* ja *Fusion* põhimõtteid samas rakenduses. Kõikide plusside juures on *Vaadin Fusion* miinuseks kesisem dokumentatsioon võrreldes *Flow* versiooniga. Selle põhjus on lihtne, tegemist on uuema rakendusega, mille dokumentatsioon on pidevalt täiendamisel [46].

3.2.2 BioOntology mõistete pärimine

Rakendus peab aitama kasutajal otsida ontoloogia mõisteid, mille vastu kaardistada relatsioonilise andmebaasi andmed. Selleks on vaja andmed kuskilt saada. Bioloogiateaduste kogukonnas on olemas, avatud bioloogilise ja biomeditsiinilise ontoloogia (*The Open Biological and Biomedical Ontologies*) kogukond edaspidi OBO, millel on ühised põhimõtted ja tavad. Enamus OBO ontoloogiatest pakuvad standardiseeritud nimesid teadusvaldkonna domeeni asjadele [47].

OBO ontoloogia nimesid on võimalik pärida kasutades BioPortal-i bioloogiateaduste repositooriumi API-d. API koosneb mõistetest, ontoloogiatest jne. API-d saab kasutada erinevate lõpp-punktide kaudu, mida võib kasutada ontoloogia kohta käiva info pärimiseks, ontoloogiliste omaduste ja mõistete otseseks otsinguks või otsinguks läbi mõiste kirjelduse. Prototüübi arendamisel läheb vaja, vaid mõistete otsest ja ontoloogia info otsingut [48].

3.2.3 Ontop API

Rakendus peab suutma transformeerida relatsioonilise andmebaasi andmed RDF-ks. Selleks on meil kasutada *Ontop API*. *Ontop* on virtuaalse graafi loomise süsteem, mille peamine funktsionaalsus on paljastada relatsioonilise andmebaasi sisu graafina. Virtuaalse graafi mõiste tähendab, et RDB andmed jäävad andmeallikasse, kuid neid on võimalik läbi *Ontop*-i ja SPARQL päringute kätte saada. See on tagatud R2RML kaardistuskeele abil, mida me vaatlesime Peatükis 2.6.1. Lisaks *Ontop* abil on võimalik automaatselt genereerida JDBC nõuetele vastava andmebaasi R2RML kaardistus. Samuti suudab *Ontop* transformeerida relatsioonilised andmed, kasutades R2RML kaardistust RDF-ks. *Ontop* kasutab enda sees RDF-ks Prototüüp rakenduses kasutame automaatse kaardistuse loomist, et seda hiljem modifitseerida R2RML API abil ning genereerida *Ontop*-iga RDF [50].

3.2.4 R2RML API

R2RML API abil on võimalik sisse lugeda R2RML kaardistust ja transformeerida selle Java klassiks. Saadud Java klassi on võimalik muuta, muutes selle omaduste väärtusi, kustutada üleliigseid või lisades uusi kaardistusi. Peale R2RML kaardistuse muutmist saab selle serialiseerida R2RML kaardistuskeelde ning kasutada seda RDB-RDF transformeerimiseks. Eelnevalt mainutud funktsionaalsust on meil vaja prototüübi tegemisel. Lisaks meie jaoks olulisele funktsionaalsusele saab *R2RML API* abil luua nullist R2RML kaardistust kasutades Java-t [51].

3.2.5 Prototüübi andmebaas

Prototüübi jaoks on vaja võimalikult lihtsalt seadistatavat andmebaasi, mida on lihtne integreerida *Vaadin* tagarakendust jooksutatavaga Spring Boot-iga. H2 andmebaas on prototüüpimiseks suurepärane kandidaat. H2 on Java-s kirjutatud relatsioonilise andmebaasi haldussüsteem. Seda saab lihtne integreerida Java rakendusse minimaalse seadistusega [49].

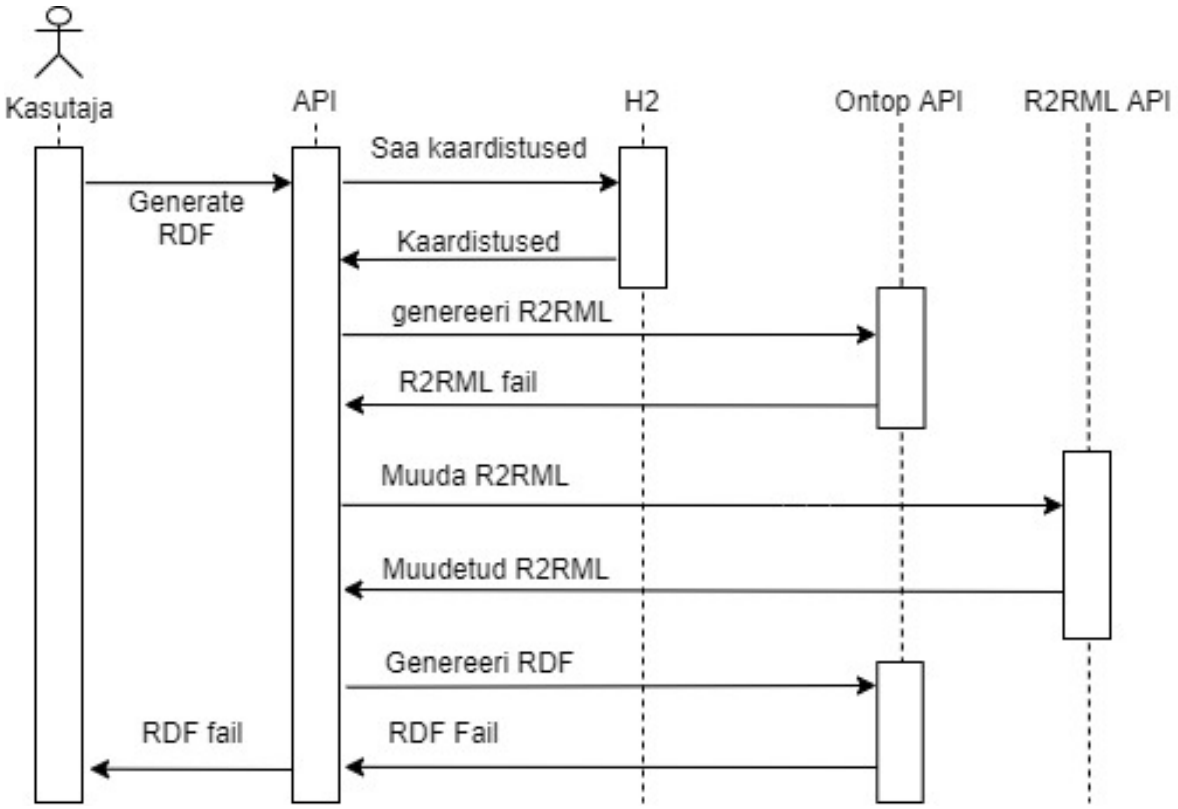
3.2.6 RDF genereerimine

Joonisel 5 on välja toodud RDF genereerimise protsess. RDF genereerimise eelduseks on loodud ühendus andmebaasiga, mille andmeid soovitakse transformeerida RDF-ks.

1. Rakendus pärib H2 andmebaasist kõik loodud kardistused.
2. Pöördudes *Ontop API* poole luuakse alus R2RML.
3. Kasutades *R2RML API*-t muudetakse alus R2RML kaardistust, mille väljundiks on muu-

detud kaardistus.

4. Genereeritakse RDF fail kasutades *Ontop API*-t.



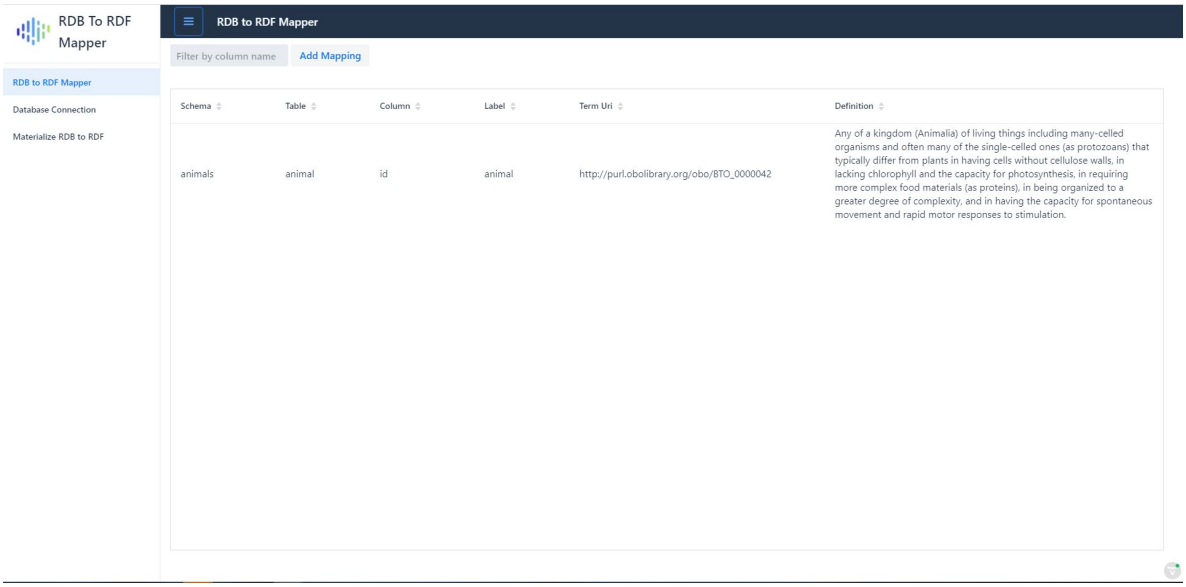
Joonis 5. RDF faili genereerimise jadadiagramm

3.3 Arenduse tulemus

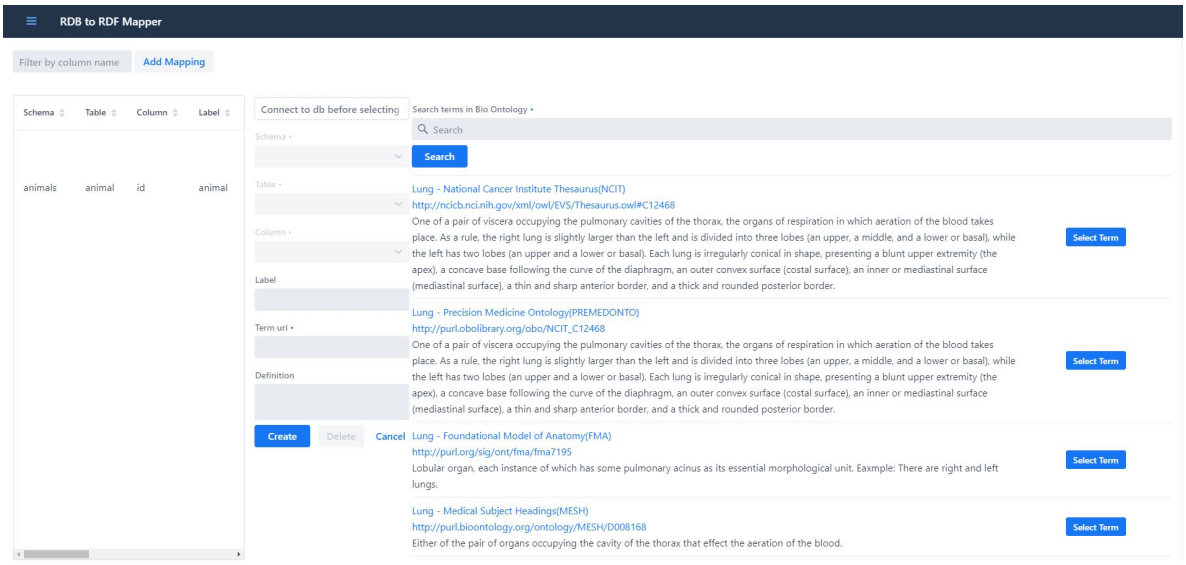
Prototüüp sai arendatud kooskõlas prototüübile esitatud nõuetega. Arendus toimus iteratsioonide kaupa. Iganädalaselt toimusid tehtud arenduse ettenäitamised Süsteembioloogia labori juhatajale Marko Vendelinile, kes andis tagasisidet tehtud arenduse osas. Pidev tagasiside võimaldas kiirelt sisse tuua muudatusi.

3.3.1 Kasutajaliidese võrdlus nõuetega

Arendatud prototüübi kasutajaliidese tulemus vastab esialgselt kokku lepitud nõuetele. Joonisel 6 on välja toodud kaardistuste vaade. Võrreldes kasutajaliidese Figma prototüübiga on lisandunud skeema tabeli veerg ning vasakus navigatsiooniribas RDB-RDF transformeerimise vaate valiku võimalus. Lisaks lisandus kaardistuste dünaamiline otsimise võimalus vasakus ülaservas. Joonisel 7 on välja toodud otsingu vaade. Vaade on arendatud vastavalt nõuetele ainukese erinevusega, et otsing aktiveerub peale nupu vajutust, mida esialgsel prototüübil ei olnud. Kõik muudatused said sisse viidud peale Marko Vendeliniga toimunud iganädalasi ettenäitamisi ning tagasiside saamist.



Joonis 6. Arendatud kasutajaliides: kaardistuste vaade



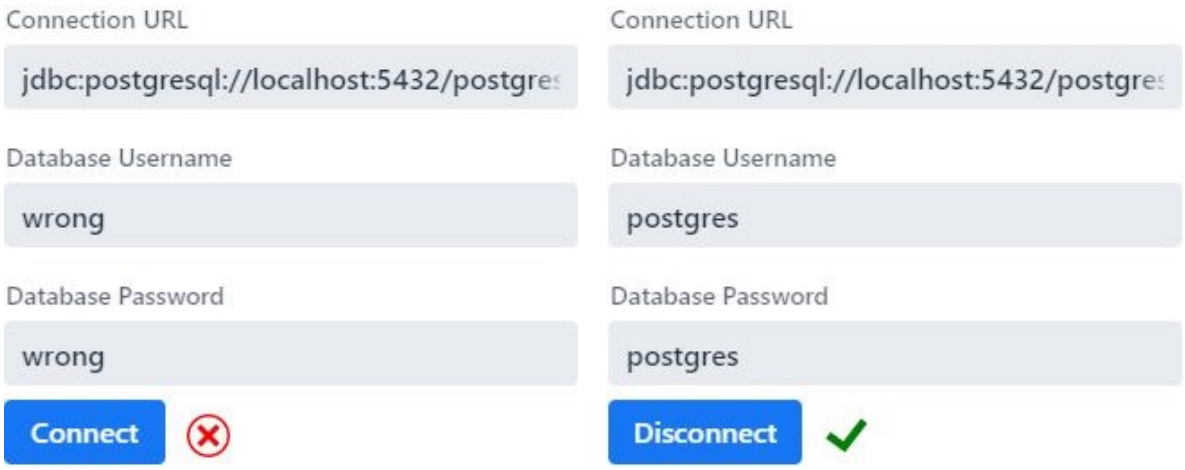
Joonis 7. Arendatud kasutajaliides: otsingu vaade

3.3.2 Funktsionaalsuse võrdlus nõuetega

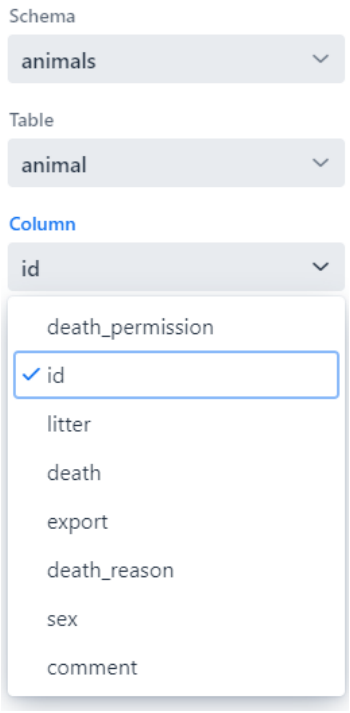
Prototüüp suudab luua ühenduse Postgresql andmebaasiga. Kasutajaliidesesse kuvatakse vastavalt ühenduse loomise staatusele, kas õnnestus või ebaõnnestus ikoon ning näidatakse sõnum ühenduse loomise staatuse kohta. See on demonstreeritud Joonisel 8.

Peale ühenduse loomist avaneb kaardistuste loomise aknas võimalus valida andmebaasi tabelite veerud, mida soovitakse annoteerida. Joonisel 9 on välja toodud, kuidas on see lahendatud kasutajaliideses. Valikus on kõik andmebaasis olevate skeemade tabelid ja nende veerud. Otsingust on eemaldatud on Postgresql haldusega seotud skeemad ja seega ka nende muu info.

Peale andmebaasiga ühenduse loomist saab kasutaja teha RDB-RDF transformatsiooni. Trans-

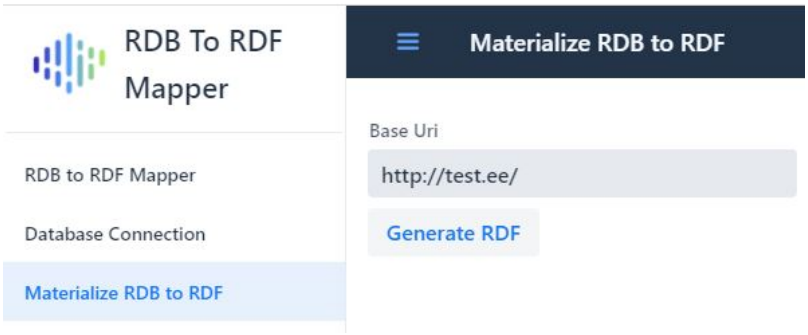


Joonis 8. Andmebaasi ühenduse loomine



Joonis 9. Kaardistatava veeru valimine

formatsiooni tegemiseks peab olema loodud ühendus ja valitud baas IRI. Baas IRI valik on oluline, kuna rakendus peab panema selle nendele RDB-RDF transformatsioonidele, millel ei ole määratud kaardistust. RDB-RDF transformatsiooni on võimalik luua ka siis kui pole kirjeldatud ühtegi kaardistust. Sellisel juhul paneb rakendus igale veerule ja tabelile baas IRI. Genereeritud RDF fail salvestatakse kasutajale arvutisse. Joonisel 10 on välja toodud RDB-RDF transformatsiooni vaade.



Joonis 10. RDB-RDF transformatsiooni vaade

3.4 RDF väljundi valideerimine

Prototüüp rakenduse poolt genereeritud RDF on Turtle serialiseerimise formaadis. Turtle sai valitud, kuna see on kasutajale kõige lihtsamini loetav ja arusaadav. Genereeritud faili süntaks sai valideeritud *IDLab Turtle Validator*-i rakendusega. Faili valideerimine õnnestub, mis tähendab et tegemist on korrektselt koostatud failiga. Kuna valiidne RDF süntaks ei garanteeri, et tegemist on RDF failiga, millega on võimalik edasi tööd teha sai kontrollitud fail jooksutades selle peale SPARQL päringut. Testimise aluseks on:

- andmebaasis on tabel “animal”,
- RDF kolmikute genereerimiseks on kasutatud kaardistust: tabeli “animal” esmatähtsa võtme vastandiks on BTO ontoloogia poolt defineeritud IRI <http://purl.obolibrary.org/obo/BTO_0000042>.
- Baas IRI-ks oli valitud <http://test.ee/>.

Protege abil jooksutades genereeritud RDF faili peale SPARQL päringut,

```
PREFIX owl: <http://www.w3.otg/2002/07/owl#>
PREFIX rdf: <http://www.w3.otg/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.otg/2000/01/rdf-schema#>

SELECT ?individual ?sex WHERE {
?individual rdf:type owl:NamedIndividual .
?individual <http://test.ee/animal#sex ?sex
} ORDER BY ?sex
```

mis pärib RDF-st kõik kolmikud, millel on predikaat <`http://test.ee/animal#sex`> ning kuvab subjekti ja selle väärtuse. Joonisel 11 on näidatud antud päringu tulemus.

?individual	?sex
<http://purl.obolibrary.org/obo/BTO_0000042=1>	F
<http://purl.obolibrary.org/obo/BTO_0000042=2>	F
<http://purl.obolibrary.org/obo/BTO_0000042=6>	F
<http://purl.obolibrary.org/obo/BTO_0000042=7>	F
<http://purl.obolibrary.org/obo/BTO_0000042=8>	F
<http://purl.obolibrary.org/obo/BTO_0000042=9>	F
<http://purl.obolibrary.org/obo/BTO_0000042=1137>	M
<http://purl.obolibrary.org/obo/BTO_0000042=3>	M
<http://purl.obolibrary.org/obo/BTO_0000042=4>	M
<http://purl.obolibrary.org/obo/BTO_0000042=5>	M

Joonis 11. SPARQL päringu tulemus

3.5 Arenduse järeldus

Prototüübi eesmärk oli kontrollida, kas kasutades analüüsis välja valitud teekide abil on võimalik transformeerida relatsioonilised andmed valiidseks RDF-ks. See eesmärk sai täidetud ja prototüüpi võib lugeda õnnestunuks. Vaatamata edukale protüübile on selge, et antud rakendus ei ole veel valmis, et seda oleks võimalik täielikus mahus kasutada andmete publitseerimiseks. Nimelt ei suuda prototüüp täita Peatükis 2.9 välja toodud nõuded, et tagada relatsioonilisest andmebaasist, FAIR printsiipidega kooskõlas , andmete mugavat publitseerimist. Selleks on vaja arendada prototüüpi edasi. Lõputöö autori hinnangul on vaja eelkõige rakendusele:

1. juurde ehitada funktsionaalsus, mis lihtsustab andmete alamhulga kätte saamist andmebaasist;
2. lisada predikaatide kaardistuse toimimine;
3. lisada *many-to-many* kaardistuste võimalus.

Nende eesmärkide saavutamine loob minimaalse töötava toote (*MVP*).

4 Kokkuvõte

Teadusandmete mugav publitseerimine, andmete edasine haldus ja nende leitavus, kättesaadavus, koostalitusvõime ja korduvkasutatavus (FAIR) on teaduses väga oluline. Selleks, et andmed oleks publitseeritavad FAIR printsiipidega kooskõlas on optimaalne viia need RDF kujule. Teadusmaailmas on enamus andmeid relatsioonilistes andmebaasides või *Excel*-i tabelites, mis teeb nende viimise RDF-i keeruliseks ja aeganõudvaks protsessiks. Bakalaureusetöö eesmärk oli luua tarkvara, mis võimaldab kasutajal eksporteerida relatsioonilises andmebaasis olevad andmed neid annoteerides ja viies kooskõlla FAIR printsiipidega. Töö käigus jõudsime selgusele, et vaadeldav probleem on liiga mahukas ja nõuab lahendamiseks palju rohkem aega, kui on

Bakalaureusetöö raames. Toetudes uutele teadmistele sai otsustatud, et arendatakse prototüüp, mis testib hüpoteesi, kas analüüsi käigus vaadeldud teegid on sobivad selleks, et teha RDB-RDF transformatsiooni lisatud annotatsioonidega. Töö tulemusena kirjeldati FAIR printsiipidega kooskõlalise publitseerimisprotsessi pudelikaelad, kasutades publitseerimise käsitsiprotsessi kaardistust ning edukalt arendati rakenduse prototüüp, mida on võimalik edasi arendada minimaalse töötava toote saavutamiseni.

Lisad

Lisa 1 - Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose Lõputöö pealkiri , mille juhendaja on Priit Järv
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

14.06.2021

Viited

[1] M. Vendelin, *IOCBIO Kinetics: An open-source software solution for analysis of data traces*, 2020 URL: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1008475 (16.05.2021)

[2] TalTech Library Open Acess, URL: https://taltech.ee/en/open-access (13.06.2021)

[3] Kinetics, URL: https://sysbio.ioc.ee/software/iocbio-kinetics/ (08.06.2021)

[4] Mark D. Wilkinson, Michel Dumontier, *The FAIR Guiding Principles for scientific data management and stewardship*, 2016 URL: https://www.nature.com/articles/sdata201618 (16.05.2021)

[5] Seek4Science, URL: https://seek4science.org/ (08.06.2021)

[6] Go Fair URL: https://www.go-fair.org/fair-principles/ (08.06.2021)

[7] Harvard Dataverse re3data, URL: https://www.re3data.org/repository/r3d100010051 (08.06.2021)

[8] FAIRDOMHub re3data, URL: https://www.re3data.org/repository/r3d100011928 (08.06.2021)

[9] M. Vendelin, *BioModels—15 years of sharing computational models in life science*, 2020 URL: https://academic.oup.com/nar/article/48/D1/D407/5614569(08.06.2021)

[10] BioStudies re3data, URL: https://www.re3data.org/repository/r3d100012627 (08.06.2021)

[11] Zenodo re3data, URL: https://www.re3data.org/repository/r3d100010468 (08.06.2021)

[12] Dryad re3data, URL: https://www.re3data.org/repository/r3d100000044 (08.06.2021)

[13] Figshare re3data, URL: https://figshare.com/ (08.06.2021)

[14] RDF Working Group, *Resource Description Framework (RDF)*, 2014 URL: https://www.w3.org/RDF/ (09.06.2021)

[15] RDF Working Group, *RDF Vocabulary Description Language 1.0: RDF Schema (RDFS)*, 2004 URL: https://www.w3.org/2001/sw/wiki/RDFS (09.06.2021)

[16] RDF Working Group, *A JSON-based Serialization for Linked Data (JSON-LD)*, 2014 URL: https://www.w3.org/2001/sw/wiki/JSON-LD

[17] RDF Working Group, *RDF in Attributes (RDFa)*, 2014 URL: https://www.w3.org/2001/sw/wiki/RDFa (09.06.2021)

[18] RDF Working Group, *Web Ontology Language (OWL)*, 2012 URL: https://www.w3.org/OWL/ (09.06.2021)

[19] O. Lassila and R. Swick, *Resource Description Framework (RDF) Model and Syntax Specification*, 1999 URL: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/ (09.06.2021)

[20] Joep Meindertsma, *What's the best RDF serialization format?*, 2019 URL: https://ontola.io/blog/rdf-serialization-formats/ (09.06.2021)

[21] RDF Working Group, *RDF 1.1 Concepts and Abstract Syntax*, 2014 URL: https://www.w3.org/TR/rdf-concepts/ (09.06.2021)

[22] RDF Working Group, *Resource Description Framework (RDF) Model and Syntax Specification*, 2014 URL: https://www.w3.org/TR/rdf-concepts/ (09.06.2021)

[23] Wikipedia, *Uniform Resource Identifier*, 2021 URL: https://en.wikipedia.org/wiki/Uniform_Resource_Identifier (09.06.2021)

[24] Wikipedia, *Internationalized Resource Identifier*, 2021 URL: https://en.wikipedia.org/wiki/Internationalized_Resource_Identifier (09.06.2021)

[25] Bikakis N., Tsinaraki C., Gioldasis N., Stavrakantonakis I., Christodoulakis S *The XML and Semantic Web Worlds: Technologies, Interoperability and Integration. A survey of the State of the Ar*, 2013 URL: http://www.dblab.ntua.gr/~bikakis/XMLSemanticWebW3CTimeline.pdf (09.06.2021)

[26] Tim Berners-Lee *Notation 3 Logic*, 2005 URL: https://www.w3.org/DesignIssues/Notation3 (09.06.2021)

[27] RDF Working Group, *RDF 1.1 Turtle*, 2014 URL: https://www.w3.org/TR/rdf-concepts/ (09.06.2021)

[28] Ontobee, URL: http://www.ontobee.org/ (09.06.2021)

[29] BioOntology, URL: http://bioportal.bioontology.org/ (09.06.2021)

[30] Ontofox, URL: http://ontofox.hegroup.org/ (09.06.2021)

[31] Courtot, Mélanie; | Gibson, Frank | Lister, Allyson L. | Malone, James | Schober, Daniel | Brinkman, Ryan R. | Ruttenberg, Alan *MIREOT: The minimum information to reference an external ontology term*, 2011 URL: https://content.iospress.com/articles/applied-ontology/ao087 (09.06.2021)

[32] Protege, URL: https://protege.stanford.edu/ (09.06.2021)

[33] Jailer, URL: http://jailer.sourceforge.net/ (09.06.2021)

[34] RDF Working Group, *R2RML: RDB to RDF Mapping Language*, 2012 URL: https://www.w3.org/TR/r2rml/ (09.06.2021)

[35] RDF Working Group, *A Direct Mapping of Relational Data to RDF*, 2012 URL: https://www.w3.org/TR/rdb-direct-mapping/ (09.06.2021)

[36] The Apache Software Foundation, *Jena Ontology API* URL: https://jena.apache.org/documentation/ontology/ (09.06.2021)

[37] James A. Overton, *OBO Tutorial*, 2017 URL: https://github.com/jamesaoverton/obo-tutorial (10.06.2021)

[38] PyRDB2RDF, URL: https://github.com/nisavid/pyrdb2rdf (10.06.2021)

[39] Stardog, URL: https://www.stardog.com/ (10.06.2021)

[40] Apache Jena, URL: https://jena.apache.org/ (10.06.2021)

[41] FU Berlin, DERI, UCB, JP Morgan Chase, AGFA Healthcare, HP Labs, Johannes Kepler Universität Linz *D2R Server: Accessing databases with SPARQL and as Linked Data* URL: http://d2rq.org/d2r-server (10.06.2021)

[42] Nikolaos Konstantinou, Dimitrios-Emmanuel Spanos, Nikos Houssos, Nikolaos Mitroup, *Exposing scholarly information as Linked Open Data: RDFizing DSpace contents*, 2014 URL: https://www.researchgate.net/publication/236881378_Exposing_Scholarly_Information_as_Linked_Open_Data_RDFizing_DSpace_contents (09.06.2021)

[43] Álvaro Sicilia, German Nemirovski, Andreas Nolle*Map-On: A web-based editor for visual*, 2014 URL: https://www.w3.org/TR/r2rml/, 2017 (10.06.2021)

[44] Karma Systems, URL: https://usc-isi-i2.github.io/karma/ (12.06.2021)

[45] Statista *Most used web frameworks among developers worldwide, as of early 2020*, 2014 URL: https://www.statista.com/statistics/1124699/worldwide-developer-survey-most-used-frameworks-web/, 2017 (11.06.2021)

[46] Vaadin, URL: https://vaadin.com/ (11.06.2021)

[47] OBO Foundry, URL: http://www.obofoundry.org/ (11.06.2021)

[48] Bioontology API, URL: http://data.bioontology.org/documentation (11.06.2021)

[49] H2 Database Engine, URL: https://www.h2database.com/html/main.html (12.06.2021)

[50] Ontop, URL: https://ontop-vkg.org/ (12.06.2021)

[51] Marius Strandhaug, *An R2RML Mapping Management API in Java*, 2014 URL: https://core.ac.uk/download/pdf/30901655.pdf (12.06.2021)